

## 1. Elección de estructuras

Todes les que estamos trabajando desde nuestra casa, sabemos de la pesadilla que es que se acumulen los platos rápidamente por estar todo el día en casa. Por eso, les docentes de Algo 2 queremos aportar para mejorar la situación. Hicimos un TAD para modelar una pileta y les pedimos a ustedes que lo diseñen.

El sistema PILETA (nos matamos con el nombre) permite dejar una pila de platos personalizada, así cada persona lava lo suyo y nadie tiene que estar lavando lo de las demás (bien individualista la cosa). Para dejar una pila de platos se llama a la operación `dejarPlatos`. Si una persona llama a `dejarPlatos` muchas veces, su pila de platos se acumula (o sea, llamar a `dejarPlatos(p, "Pepe", [1,2])` y después a `dejarPlatos(p, "Pepe", [3,4])` es equivalente a llamar a `dejarPlatos(p, "Pepe", [3,4,1,2])`, siempre lo primero es lo último que se agrega). Los nombres de las personas tienen longitud acotada (se puede asumir que la operación de copiar o guardar un nombre tiene tiempo constante)

Como no queremos que ninguna pila crezca demasiado, sólo puede lavar quien tenga la pila más numerosa. Si hay varias personas que tienen la pila de igual tamaño, se determina quién lava de alguna manera. También nos piden saber quien es le que más lavó históricamente y quien es le que tiene más vajilla por lavar actualmente.

A continuación les dejamos la especificación del problema:

**TAD** VAJILLA ES NAT  
**TAD** PERSONA ES STRING

**TAD** PILETA

I

**géneros**      pileta

**observadores básicos**

personasConPila : pileta  $\rightarrow$  conj(persona)

porLavar : pileta  $p \times$  persona  $per \rightarrow$  secu(vajilla)  $\{per \in \text{personasConPila}(p)\}$

cuántoLavó : pileta  $p \times$  persona  $per \rightarrow$  nat

**generadores**

nueva :  $\rightarrow$  pileta

dejarPlatos : pileta  $p \times$  persona  $per \times$  secu(vajilla)  $vs \rightarrow$  pileta  
 $\{(\forall per' : \text{persona})(\forall v : \text{vajilla}) per' \in \text{personasConPila}(p) \wedge \text{está?}(v, vs) \Rightarrow_L \neg \text{está?}(v, \text{porLavar}(p, per'))\}$

lavar : pileta  $\rightarrow$  pileta

**otras operaciones**

elQueMásLavó : pileta  $p \rightarrow$  persona  $\{(\exists per : \text{persona}) \text{cuántoLavó}(per) > 0\}$

elQueMásTieneParaLavar : pileta  $p \rightarrow$  persona  $\{\neg \text{vacío?}(\text{personasConPila}(p))\}$

**Fin TAD**

Dadas las siguientes operaciones con las complejidades temporales de peor caso indicadas (teniendo en cuenta que  $P$  es el conjunto de personas que pasaron por el sistema y  $s$  es la vajilla que se agrega a la pila):

- **PERSONASCONPILA**(**in** p: pileta, **out** ps: conj(persona))  
Devuelve las personas que tengan platos pendientes de lavar.  
Complejidad:  $O(1)$
- **PORLAVAR**(**in** p: pileta, **in** per: persona, **out** vs: secu(vajilla))  
Devuelve los platos por lavar de la persona per.  
Complejidad:  $O(\log(\#P))$
- **DEJARPLATOS**(**in/out** p: pileta, **in** per: persona, **in** s: secu(vajilla))  
Agrega una pila de platos a los de esa persona.  
Complejidad:  $O(\log(\#P) + \text{len}(s))$
- **LAVAR**(**in/out** p: pileta)  
Elimina la primer vajilla de la secuencia más larga.  
Complejidad:  $O(\log(\#P))$

- **ELQUEMASLAVO**(in p: pileta)  $\rightarrow$  per : persona  
Devuelve la persona que más vajillas lavó en total.  
Complejidad:  $O(1)$
- **ELQUEMASTIENEPARALAVAR**(in p: pileta)  $\rightarrow$  per : persona  
Devuelve la persona que más vajillas tiene actualmente por lavar.  
Complejidad:  $O(1)$

Se pide:

1. Dar una estructura de representación para un módulo **PILETA** (que provee las operaciones mencionadas), explicando detalladamente qué información se guarda en cada parte, las relaciones entre las partes, y las estructuras de datos subyacentes.
2. Justificar detalladamente de qué manera es posible implementar cada una de las operaciones para cumplir con las complejidades pedidas. Escribir el algoritmo para la operación **LAVAR**.

Para la resolución del ejercicio no está permitido utilizar módulos implementados con tabla hash de base. Esto se debe a dos motivos: uno porque queremos que combinen el resto de las estructuras vistas, y otro porque los peores casos de la tabla hash exceden los que se piden en los ejercicios (por ejemplo, duplicar el espacio en una tabla cuesta  $O(n)$  en el peor caso).

Donde se menciona 'secu' se puede usar alguno de los módulos vistos que se explican con el TAD Secuencia, de acuerdo a lo que está en el apunte de Módulos Básicos.